Pointer Networks: A Unified Approach to Extracting German Opinions

Julia Wunderle and **Jan Pfister** and **Andreas Hotho**
Julius-Maximilians-Universität Würzburg
Computer Science Chair X: Data Science
{lastname}@informatik.uni-wuerzburg.de

Abstract

Transformer-based pointer networks currently represent the state of the art for English aspect-based sentiment analysis. Inspired by their performance in extracting structured sentiment information from text, we aim to transfer this success to the German language. For evaluation we use the GermEval shared task on "Aspect-based Sentiment in Social Media Customer Feedback", as it consists of four subtasks: (A) Relevance Classification, (B) Document-level Polarity, (C) Aspect-level Polarity, and (D) Opinion Target Extraction (Wojatzki et al., 2017). We follow the intuition of the English approach by training a single model to solve all related subtasks at once. Therefore, the subtasks are formulated as a single unified index generation problem, enabling the model to solve all four subtasks simultaneously. We find that solving all four subtasks at once only has a minimal impact on the overall performance of our model. Consequently, we closely match or outperform all previous approaches despite them training subtask-specific models.

1 Introduction

Explicit customer feedback is an extremely valuable source for understanding the needs of customers and improving products and services accordingly, while being available in a large amount on the Internet in unstructured form. It is necessary to be able to aggregate and analyze the feedback in a comprehensive way, to understand the wide variety of opinions and sentiments expressed. Ideally, the feedback has to be extracted in a fine-grained but easily understandable way. The main interests are, of course, voiced opinions and their aspects that determine whether and why exactly, e.g. a review is positive or negative. Consequently, to break down a long review to its core information, tuples of opinion terms and their associated sentiment have to be extracted. This structured span-based extraction process, called "Opinion Extraction", can easily be solved by pointer networks. Specifically, BARTABSA (Yan et al., 2021) is a sequence-to-sequence pointer model, which predicts a sequence of class tokens and pointers to token indices of the input text. Current research developments related to pointer networks and aspect-based sentiment analysis do include multilingual approaches (R et al., 2022; Pfister et al., 2022) but notably until now not German. Thus, the research gap arises on how recent advances in structured sentiment prediction can be leveraged for German Opinion Extraction. Despite this gap, for German evaluation there exists a comparably large data set introduced by the GermEval 2017 Task that contains customer feedback about "Deutsche Bahn". To this end, four subtasks were formulated and, while all are related to the analysis of customer feedback, each subtask focuses on a different level of information classification and extraction. In order of increasing complexity, the formulated tasks are (A) Relevance Classification, (B) Document-level Polarity, (C) Aspect-level Polarity (D) Opinion Target Extraction (Wojatzki et al., 2017). Thus, to solve all tasks successfully, a model needs to not only classify the entire document itself but also extract and label all relevant spans correctly. In contrast to existing approaches, we leverage a pointer network to solve all of these subtasks with the same model simultaneously, which we find to sometimes even increase performance over a model specialized on a subset of the tasks. In summary our main contributions are: (i) Formulating all GermEval 2017 subtasks as a single unified index generation problem, (ii) thereby introducing document-level classes next to sentiment spans to the BARTABSA approach. (iii) Extensively evaluating various German-capable transformer encoder-decoder basemodels on this German language task. (iv) We show that our model outperforms or closely matches all previously existing approaches while solving all tasks at once.

Proceedings of the 19th Conference on Natural Language Processing (KONVENS 2023), pages 127–138
September 18–22, 2023.

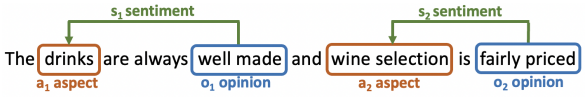

Figure 1: Exemplary input sentence for aspect-based sentiment analysis, annotated with aspects, opinions and sentiments (Yan et al., 2021).

2 Preliminaries & Related Work

2.1 Seq2Seq-Transformers

Transformer models consisting of an encoder and decoder are commonly referred to as *Seq2Seq* models, as the encoder generates an intermediate representation of the input sequence using which the decoder then generates the output sequence. In our experiments, we compare different Seq2Seq models with each other.

First, *BART* was pretrained as a denoising autoencoder (Lewis et al., 2020). Here denoising refers to the training process, in which a noised/-masked text sequence is given as input and the model is trained to reproduce the original sequence as output. The model is applicable to a wide range of tasks such as sequence classification, token classification, sequence generation, or machine translation. We also explore a BART model fine-tuned on the MNLI task as inspired by R et al. (2022) and one fine-tuned on the German MLSUM dataset (Scialom et al., 2020). Furthermore, we use *mBART50* which was trained in translating between 50 languages, including German (Tang et al., 2020). Lastly, *M2M-100* is a Many-to-Many multilingual Seq2Seq model trained on sentence pairs to translate between any pair of 100 languages, including German (Fan et al., 2020).

2.2 Aspect-Based Sentiment Analysis

The goal of *Aspect-based Sentiment Analysis (ABSA)* is, given a sentence containing expressed opinions, to extract explicitly voiced opinions, each consisting of an aspect term, its opinion term and corresponding sentiment polarity (see Figure 1 for example). Thereby, the aspect terms are the target to which the opinion terms refer, and thus express the polarity of the sentiment.

This task consists of two types of subtasks: extraction and classification. Here extraction refers to extracting and annotating the span of terms (a_1, a_2, o_1, o_2 in Figure 1), while classification describes the prediction of characteristics of this relationship, e.g. sentiment polarities (s_1, s_2).

2.3 BARTABSA Pointer Network

The BARTABSA pointer framework introduced by Yan et al. (2021) proposes a unified solution to solve various predefined ABSA subtasks. This comes with a substantial performance gain over comparable well-performing baselines, including BERT-based approaches. To achieve this, they reformulate all subtasks as a unified generative task, meaning that every subtask is defined as a sequence of pointers to indices in the source sequence and sentiment class tokens. To predict pointers to indices of the source sequence, they implement a pointer network, which uses BART as a backbone. Unlike a regular transformer, pointer networks do not output a probability distribution over a vocabulary of fixed size, but instead a distribution over tokens of the input sequence.

The model works by first generating an input representation $H^e \in \mathbb{R}^{n \times d}$ from its encoder. Here, d denotes the embedding size and n the number of tokens in the input sequence. This H^e is used by the decoder to autoregressively generate the target sequence. In every timestep, it takes H^e and previously generated tokens $Y_{<t}$ as input and returns a vector $H^d \in \mathbb{R}^d$. To obtain a token probability distribution P^X over the input sequence, the following calculations are performed: Both - the input sequence X, and the list of class tokens C - get embedded by the token embedding layer of the model, resulting in the embedding vectors $E^X \in \mathbb{R}^{n \times d}$ and $E^C \in \mathbb{R}^{l \times d}$, where l denotes the number of class tokens in the vocabulary. Next, a weighted average is calculated between the encoder output H^e and the embedded input sequence E^X, to obtain a new representation $\bar{H}^e \in \mathbb{R}^{n \times d}$.

$$\bar{H}^e = \alpha \text{MLP}(H^e) + (1 - \alpha) E^X \quad (1)$$

Before calculating this weighted average, H^e gets processed by a multilayer perceptron (MLP). Finally, the pointer distribution over the input tokens $P^X \in \mathbb{R}^{n+l}$ is calculated, by the softmax over the concatenation of $\bar{H}^e$ and E^C times H^d.

$$P^X = \text{Softmax}([\bar{H}^e \parallel E^C] H^d) \quad (2)$$

In order to use the list of previous predictions $Y_{<t}$ as autoregressive input, all pointers are replaced by their respective token they are pointing to from the input sequence before feeding them to the decoder.

2.4 GermEval 2017: Aspect-Based Sentiment in Social Media Customer Feedback

The GermEval 2017 task is a shared task on analyzing customer reviews and news related to "Deutsche Bahn" and provides an annotated data set of 26 209 documents for training and evaluation. The shared task consists of four subtasks to be tackled individually (Wojatzki et al., 2017).

(A) Relevance Classification: The goal of this subtask is to classify a document as relevant (true) or irrelevant (false) for Deutsche Bahn.

(B) Document-Level Polarity: This subtask is about concluding whether the entire customer review is overall positive, neutral, or negative.

(C) Aspect-Level Polarity: Subtask C involves the identification of all categories mentioned in the document and their associated polarity.

(D) Opinion Target Extraction: The goal of subtask D is to identify the exact term(s) in the document matching the categories and their polarity from subtask C. Each term is predicted as a span in the document, and a single span can be associated to multiple categories.

2.5 Data Set

The provided data was collected using web scraping with a list of query terms, from May 2015 to June 2016, thus covering various seasonal and everyday problems such as holidays or strikes (Wojatzki et al., 2017).

In the following, we take a close look at the training data and list common properties that we found. In general, the data is divided into two main categories: irrelevant and relevant to the topic of "Deutsche Bahn". Irrelevant documents do not contain annotated opinions and the sentiment is always set to "neutral". Relevant data can be further split into two subcategories: Some documents contain clearly expressed opinions, which are annotated accordingly, while others are topically relevant but do not contain any concrete opinions. In the latter case, the opinion term, represented by a span in the source document, is set to "NULL". Thus the data can be divided into the following three types: 1. irrelevant 2. relevant without annotated opinion spans 3. relevant with annotated opinion spans.

The GermEval subtasks C and D require classifying the opinion terms according to suitable categories. In total, there are 20 main categories and

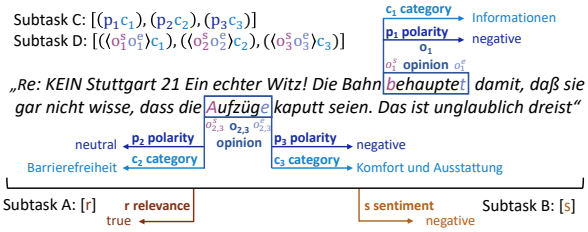

Figure 2: Example document containing labels for all four subtasks and our sequential encoding for each.

up to 54 subcategories used to categorize the opinion terms. For the purpose of the shared task, it is sufficient to predict the correct main categories, as the subcategories are not taken into account. Notably, a single opinion term can be associated with multiple categories (opinions 2 and 3 in Figure 2).

3 Methodology

Inspired by the performance of pointer networks in English aspect-based sentiment, we adapt and extend the BARTABSA framework (2021) aiming to solve the four subtasks introduced previously (Section 2.4) - at once and in German.

3.1 Formulating the Subtasks as a Single Sequence-to-Sequence Task

In order to predict all subtasks at once, the output sequence of our model needs to take into account all spans and labels required to extract the four subtasks. If we are able to model all four subtasks as a single sequence, we can consequently train a model to predict this sequence. This sequential representation has to be unambiguous, so that the output of the model is always correctly interpretable. In the following, we define our prediction targets as a sequence consisting of index pointers and class tokens, as shown in Figure 2 and Table 1.

Document-Level Classification

The first two subtasks are document-level classification tasks, which are represented by the first two rows in Table 1. The output for both tasks can be modeled by only a single special token, each specifying the class the document belongs to. For subtask A we define $r \in \{\text{true}, \text{false}\}$ as the relevance class token, indicating whether the document is relevant to the topic of "Deutsche Bahn" or not, while for subtask B we define $s \in \{\text{positive}, \text{negative}, \text{neutral}\}$ as the sentiment class token, indicating the overall sentiment of the entire input document.

Table 1: Representation of the target sequences required to solve all four GermEval subtasks first individually and then together. Here i represents the i^{th} category and opinion span present and | depicts the separator token.

Task	Target Sequence Representation
A	$[r]$
B	$[s]$
C	$[c_1, p_1 \mid \ldots \mid c_i, p_i \mid \ldots]$
D	$[o^s_1, o^e_1, c_1 \mid \ldots \mid o^s_i, o^e_i, c_i \mid \ldots]$
Comb.	$[r, s \mid o^s_1, o^e_1, c_1, p_1 \mid \ldots \mid o^s_i, o^e_i, c_i, p_i \mid \ldots]$

Category & Span Prediction

For subtask C a combination of two classes has to be predicted: a category implicitly or explicitly rated in the input document and additionally the expressed sentiment for each category. Therefore, we define $c \in \{\text{Allgemein}, \text{Zugfahrt}, \text{Ticketkauf}, \ldots\}$ as the category and $p \in \{\text{positive}, \text{negative}, \text{neutral}\}$ as the class token associated with the category. A document can contain several different opinions, consequently a category-polarity pair has to be predicted for each mentioned category. We enumerate and predict these pairs in the order they occur in the text input, which is why we introduce i representing the i^{th} pair, associated to the i^{th} mentioned category. For subtask D, in addition to the category c and polarity p the matching opinion term o has to be predicted, where applicable. Therefore, we introduce pointer indices corresponding to the start and end index of the target opinion term in the source sequence, which we indicate with o using superscript to mark the start^s and end^e index token. Again, i represents the i^{th} term, and again we encode the opinion terms in the order in which they appear in the text. It is important to note that a single opinion span can be associated with multiple categories and polarities (see Figure 2).

Finally, to solve all four subtasks at once, we string together the four target sequences into one by concatenating subtasks A and B, while interleaving the matching parts of subtasks C and D (Table 1). In doing so, we must carefully consider the data types we identified in Section 2.5. We distinguish between these three mentioned kinds of data types in the process, to achieve a natural encoding for all data points. In the following, we lay out the three encodings for the different data types ordered by increasing complexity.

Document irrelevant to "Deutsche Bahn"

{"text": "RT @DLR_next: Ach ja: Sie dürfen jetzt das Alu-Hütchen wieder absetzen. Zu unserer eigenen Überraschung hatten wir die Asteroiden-Bahn korr", "relevance": "false", "sentiment": "neutral"}

Data points labeled as not relevant for the topic of "Deutsche Bahn" are characterized by their target for subtask A being "not relevant" and sentiment for subtask B being "neutral". Furthermore, these data points do not contain annotated categories, polarities, or opinion terms for tasks C and D. Consequently, the target sequence contains the following information: relevance and sentiment, which gets encoded as two tokens: [BOS, false, neutral, SEP, EOS].

Document relevant but without Opinion Terms

{"text": "@DB_Bahn Gibts denn ne Ersatzfahrt oder so?!", "relevance": "true", "sentiment": "neutral", "opinions": [{"category": "Allgemein", "polarity": "neutral"}]}

Next we address data points which are relevant and thus have categories and polarities annotated but do not contain opinion terms. We extend our existing encoding scheme by appending a list of categories and polarity tuples to the target sequence: [BOS, true, neutral, SEP, Allgemein, neutral, SEP, EOS]. The sequence now contains the information: relevance, sentiment, category and polarity.

Document relevant and contains Opinion Terms

{"text": "Juhu Weichen Störung! Ich liebe die Bahn...Nicht -.-", "relevance": "true", "sentiment": "negative", "opinions": [{ "category": "Allgemein", "polarity": "negative"}, {"category": "Unregelmässigkeiten", "polarity": "negative", "from": 1, "to": 2, "term": ["Weichen", "Störung"]}]}

Finally, we add the ability to encode the spans that represent opinion terms in our target sequence. As indicated in Table 1, one document can be annotated with multiple categories or opinion terms (spans). These spans are associated with the category we implemented above, and consequently we concatenate these to their respective category. The sequence thus has to contain the document's relevance and sentiment as well as a list of $\text{opinion}_{\text{start}}$, $\text{opinion}_{\text{end}}$, category and polarity. For above example we define this target sequence: [BOS, true, negative, SEP, Allgemein, negative, SEP, 1, 2, Unregelmässigkeiten, negative, SEP, EOS]. Following the BARTABSA encoding, we first predict the span and the associated category afterwards.

3.2 Pointer Network

Architecturally we keep the model introduced in BARTABSA (Section 2.3) unchanged. To enable it to predict our previously defined target sequence, we need to extend the special token vocabulary of our model, as the task at hand includes document-level classes as well as 20 categories, all of which need to be predicted by our model. The vocabulary has to contain the following special tokens: (i) BOS, EOS, PAD, SEP (ii) two tokens for document relevance: true, false (iii) three tokens for sentiment and category polarity: positive, negative, neutral (iv) 20 tokens for the categories (Allgemein, Zugfahrt, Ticketkauf, etc.). At every decoding step, the pointer network either predicts a pointer to a token index of the source sequence, or a class special token. Conceptionally, these special tokens are assigned to the lowest available ids, so the first 29 tokens (4+2+3+20) are special tokens. Predictions larger than this offset are interpreted as pointers to indices of tokens in the source sequence. The target sequence is created by converting all tokens to ids and then adding this constant offset of 29 to all index pointers to the source sequence. Thus our previous example of [BOS, true, negative, SEP, Allgemein, negative, SEP, 1, 2, Unregelmässigkeiten, negative, SEP, EOS] becomes [0, 4, 7, 3, 9, 7, 3, 30, 31, 14, 7, 3, 2]. The model is then trained to predict this sequence of special tokens and indices for each input document. Of course, before converting the predicted index pointers back to spans, this constant offset is subtracted again.

3.3 Handling Encoding Issues

Inputs longer than the model's context size are truncated such that during evaluation twenty data points of the synchronic test set and eleven of the diachronic test set could not be encoded in its entirety. During evaluation we set fallback defaults of relevance=true for subtask A and sentiment=neutral for subtask B for data points, where the model fails to predict either of these tasks. Both values are the respective majority classes in the training set. This is necessary to evaluate our results, as the original evaluation binary provided by the task hosts cannot handle missing values. For subtasks C and D, no fallback predictions are required or set.

4 Experiments

4.1 Data Set

To assess the robustness of the participating systems, two test sets were introduced. In addition to the "synchronic test set" (Test_{syn}), a "diachronic test set" (Test_{dia}) is provided, consisting of documents from a different time frame: November 2016 to January 2017 (Wojatzki et al., 2017). In total, the data set consists of 26 209 German messages across all splits (Train: 19 432 (of which 3231 irrelevant for "Deutsche Bahn"), Dev: 2369, Test_{syn}: 2566, Test_{dia}: 1842), annotated with the document id, relevance, and sentiment, as well as the opinion terms including exact spans, its sentiment, and category.

4.2 Evaluation & Metric

For evaluation, we use the original GermEval evaluation script, which compares the predicted results considering the micro-averaged F_1-score (Wojatzki et al., 2017). For subtasks A and B the F_1-score is reported, while for subtask C, the task hosts distinguish two types of metrics: (C1) Only the category has to match the ground truth. (C2) In addition to the category, the polarity must be predicted correctly. Furthermore, for subtask D also two types of results are differentiated: (D1) Exact result: The "from" and "to" tags have to exactly match the ground truth. (D2) Overlap result: The "from" and "to" tags can deviate from the ground truth by +/- 1 at the word level. We report our performance for all task metrics.

4.3 Hyperparameter Search

In order to improve the performance of our model, we perform an extensive training hyperparameter search. This includes exploring various ways to encode our targets, hoping to gain a better understanding of how the models performance is influenced by the different parameters. To gain a better understanding of how well each base model works for this German task, we perform a grid search, examining the hyperparameters listed in Table 8, resulting in 300 combinations of parameters. This enables us to find the best working model for German by comparing the performance of all available models against each other, while also finding the best hyperparameter combination for each of them. For batch size, epochs, and learning rate, we decide to search and analyze parameters close to the values proposed by Yan et al. (2021) and keep AdamW.

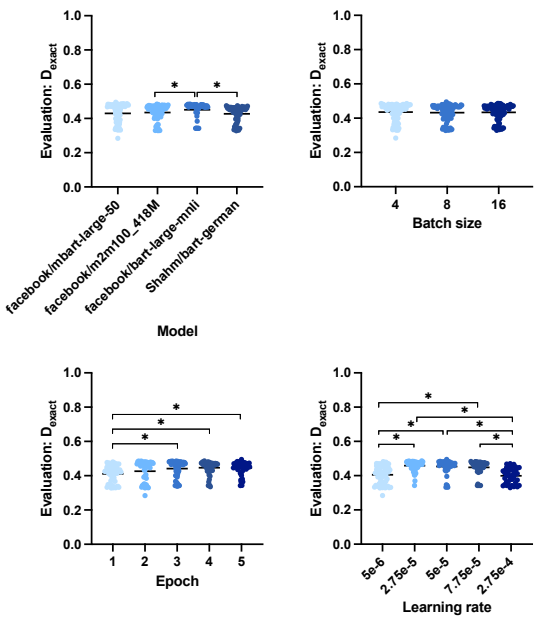

Figure 3: Results for subtask D1, aggregated for each hyperparameter configuration. The hyperparameters are listed on the x-axis and the results for D1 on the y-axis.

5 Evaluation and Results

First we evaluate the findings of our hyperparameter search, before we analyze the impact of different prediction orders. Afterwards, we compare our performance against previous approaches.

5.1 Hyperparameter Study

We systematically evaluate the impact of each hyperparameter on the overall performance of our approach. To identify statistically significant differences between hyperparameter combinations, we performed the Kruskal-Wallis test (Corder and Foreman, 2014) and corrected for multiple comparisons using post hoc Dunn's test (Dunn, 1964). Large learning rates can result in training instabilities for some hyperparameter combinations, so the exploration of the largest learning rate (5e-4) was canceled early after no valid run could be conducted. Thus, we do not have paired data points for all learning rates.

To this end, we consider $p < 0.05$ as a statistically significant difference and mark it with an $*$ in Figure 3 and Appendix A. We use a black line to represent the average values. In Figure 3 we representatively plot the performance on the validation set for subtask D1. The results for the other subtasks are similar, and their plots can be found in Appendix A. To prevent overfitting and ensure that the test set remains independent for unbiased model evaluation, the hyperparameter study is performed using the validation split.

Hyperparameter: Base Model Figure 3 illustrates that the choice of the base model can have a significant impact on the overall performance. Here, we refer to the evaluated base models by their HuggingFace identifier. We find that "facebook/bart-large-mnli" performs best, achieving significantly better results than "facebook/m2m100_418M" and "Shahm/bart-german". The difference between "facebook/bart-large-mnli" and "facebook/mbart-large-50" is minimal.

Hyperparameter: Batch Size Noticeably, the batch size has a smaller effect on the performance of the model and does not show significant differences between the different configurations.

Hyperparameter: Epochs Inspecting the number of epochs shows that training just one epoch is, as expected, statistically worse than training for three, four, or five epochs. Nevertheless, increasing the number of epochs to more than two does result in slight but not significant improvements.

Hyperparameter: Learning Rate Regarding the learning rate a clear performance deterioration can be observed starting from a learning rate of 7.75e-5 and becomes significantly worse when the learning rate increases further. The learning rate suggested by Yan et al. (2021) of 5e-5 also achieves statistically significant better results than 5e-6. We consequently identify a learning rate of 5e-5 to be the best option over all tasks.

Selected Hyperparameters After conducting this hyperparameter search, we select the configuration that ranks best across the most subtasks on the validation split. In doing so, we identify "facebook/mbart-large-50" as the best model with a learning rate of 5e-5, when training for 5 epochs with a batch size of 8. As listed in Table 9 we find that this configuration is among the top-3 combinations for subtasks C and D for both metrics each. Therefore this configuration will be used for all following analyses.

5.2 Performance on Subtasks C and D

In order to train the model to solve all subtasks at once, we have to consider all data points in the training set. In particular, this includes training on a large number of data points that are irrelevant

Table 2: Comparison of results achieved when predicting all subtasks, versus only subtasks C and D. Furthermore different permutations for subtasks C and D are evaluated. First column matches Section 3.

Ordering	r, s, o^s, o^e, c, p	o^s, o^e, c, p	o^s, o^e, p, c	c, p, o^s, o^e	p, c, o^s, o^e
C1	.657	.676	.688	**.695**	.693
C2	.543	.559	.556	.569	**.581**
D1	.476	.489	.468	.494	**.500**
D2	.500	.512	.490	.519	**.523**

Table 3: Comparison of results on the validation set for all subtasks using the best hyperparameter combination and only changing the order of the target sequence

Ordering	r, s, o^s, o^e, c, p	s, r, o^s, o^e, c, p	s, r, p, c, o^s, o^e	r, s, p, c, o^s, o^e	p, c, o^s, o^e, s, r	p, c, o^s, o^e, r, s
A	.958	.954	**.959**	**.959**	.955	.957
B	.813	.823	**.827**	.822	.824	.823
C1	.657	.662	.677	.677	.673	**.684**
C2	.543	.554	.560	**.563**	.559	.562
D1	.476	.487	.486	.490	**.498**	.497
D2	.500	.507	.510	.509	.518	**.522**

for subtasks B, C and D, as these data points are only required for subtask A (Section 4.1). As all previous approaches train separate models for each subtask, this leads us to investigate how the performance of our model changes, when dropping all irrelevant data points from the training set and training only on the two closely related and most complex tasks: subtasks C and D. Table 2 lists the results achieved, while training our model only on subtasks C and D in the first section. Comparing the first two columns, it is noticeable that leaving out subtasks A and B slightly improves the performance on subtasks C and D, although the impact on subtask C seems to be slightly larger. We deduce that additionally predicting subtasks A and B, and thus even training on a significant amount of off-topic data, does not impact performance strongly.

Table 4: Comparison of existing approaches for subtask A. Best in **bold**, second underlined.

Team Subtask A	**Test$_{syn}$**	**Test$_{dia}$**
Wojatzki et al.	0.852	0.868
Sayyed et al.	0.903	0.906
Hövelmann and Friedrich	0.899	0.897
Aßenmacher et al.	**0.957**	**0.948**
Our (A,B,C&D)	0.953	0.943

5.3 Order of Prediction

In preliminary experiments, we found slight differences in performance when changing the order of the target sequence introduced in Section 3. Therefore, we systematically evaluate the impact of this order and list the results for different prediction orders of subtasks C and D in Table 2. We find that our proposed order in Section 3 overall, scores rather low among all possible permutations and that moving the predictions for subtask C in front of subtask D slightly improves the results.

Consequently, in Table 3 we take the best prediction order for subtasks C and D and analyze it in combination with different permutations of subtasks A and B. We find that the overall differences are small, but predicting subtasks A and B last (p, c, o^s, o^e, r, s) achieves the best results considering the mean over all subtasks. Thus, this is the prediction order we fix for further evaluation.

5.4 Results on the Test Set

Previous analyses were conducted on the validation split, while the following comparisons against existing approaches are carried out on the test set. We also examined the impact of different seeds on the performance, to get insights into robustness and reproducibility (Table 10). Notably, in Section 5.1 we selected a combination that performs better on subtasks C and D than on A and B.

(A) Relevance Classification:

For subtask A (Table 4), our approach outperforms all original participants and the system provided by the organizers (Sayyed et al., 2017; Hövelmann and Friedrich, 2017; Wojatzki et al., 2017). With the recent approach by Aßenmacher et al. (2021) achieving the best results, our model comes second. Our model predicts all four subtasks at once while remaining competitive with Aßenmacher et al. who trained a separate model for each subtask.

Table 5: Comparison of existing approaches for subtask B. Best in **bold**, second best underlined.

Team Subtask B	**Test$_{syn}$**	**Test$_{dia}$**
Wojatzki et al.	0.667	0.694
Naderalvojoud et al.	0.749	0.736
Hövelmann and Friedrich	0.748	0.742
Aßenmacher et al.	0.807	0.800
Our (A,B,C&D)	**0.815**	**0.811**

Table 6: Comparison of existing approaches for subtask C. Best in **bold**, second best underlined.

Team Subtask C	**C1$_{syn}$**	**C2$_{syn}$**	**C1$_{dia}$**	**C2$_{dia}$**
Wojatzki et al.	0.481	0.322	0.495	0.389
Lee et al.	0.482	0.354	-	-
Mishra et al.	0.421	0.349	0.460	0.401
Aßenmacher et al.	0.761	0.655	0.791	0.689
↪ reevaluated	0.614	0.475	0.649	0.493
Our (C&D)	0.624	**0.514**	**0.657**	**0.553**
Our (A,B,C&D)	**0.632**	0.510	0.634	0.535

(B) Document-Level Polarity:

For subtask B (Table 5) our model not only outperforms all original participants (Wojatzki et al., 2017; Naderalvojoud et al., 2017; Hövelmann and Friedrich, 2017) but also the newer approach by Aßenmacher et al. (2021) while solving all subtasks at once. Again, the newer LLM-based approaches clearly perform better than previous models.

(C) Aspect-Level Polarity:

For subtask C (Table 6), our approach again outperforms all original participants (Wojatzki et al., 2017; Lee et al., 2017; Mishra et al., 2017). While Aßenmacher et al. (2021) report better results for subtask C, it should be noted that their scores are calculated using a custom reimplementation of the evaluation metric. Consequently, we reevaluated their outputs using the original GermEval metric and achieved different results for subtask C, as we detail in Appendix C.1. We assume that this discrepancy results from different calculations of the average (micro vs. macro). Nevertheless, we list both values in Table 6, the result of their custom metric in gray, and the result we calculated using the original metric in black. We report the results achieved by the best model trained from each: Table 2 and 3. Our model trained only on subtasks C and D outperforms all other previous approaches for all reported results, including our model trained on all subtasks in three out of four cases. Nevertheless, we find that our model trained on all subtasks is able to outperform our model, which specializes in subtasks C and D once, and even outperforms all previous approaches in three out of four cases.

Table 7: Comparison of existing approaches subtask D. Best in **bold**, second best underlined.

Team Subtask D	**D1$_{syn}$**	**D2$_{syn}$**	**D1$_{dia}$**	**D2$_{dia}$**
Wojatzki et al.	0.170	0.237	0.216	0.271
Mishra et al.	0.220	0.221	0.281	0.282
Lee et al.	0.203	0.348	-	-
Aßenmacher et al.	0.515	0.523	0.518	0.533
Our (C&D)	0.404	0.430	0.442	0.471
Our (A,B,C&D)	**0.415**	**0.440**	**0.448**	**0.479**

(D) Opinion Target Extraction:

For subtask D (Table 7), our approach again comfortably outperforms all previous approaches (Wojatzki et al., 2017; Lee et al., 2017; Mishra et al., 2017). Despite close contact with the authors of Aßenmacher et al. (2021), we were unable to generate reportable results for their approach using the original evaluation script, as we assume that their approach suffers from a preprocessing bug (C.2). Interestingly, again we find that our model trained on all subtasks is able to outperform our model trained only on subtasks C and D, verifying our intuition of training a single model for all subtasks.

6 Conclusion

In this work, we proposed the first approach that is able to solve all four subtasks of the GermEval 2017 shared task simultaneously. To achieve this goal, we used a pointer network to sequentially predict a single, unified target sequence that encodes all subtasks. We conducted an extensive hyperparameter search and thoroughly evaluated different configurations and orders of prediction. In doing so, we find that predicting all subtasks at once does not negatively impact the overall performance of our model and on the test set can even result in a performance increase, verifying our strategy of unifying all subtasks. Consequently, although our model solves all subtasks at once, it outperforms or closely matches all previous approaches on both test sets. For all subtasks we find that our results on the synchronic and dedicated diachronic test sets are very similar, indicating robustness.

7 Limitations

Certain limitations of our approach should be considered. Our approach to a large degree assumes that not only multilingual base models, but even English-only base models are effectively applicable to the German language. This may not translate well to more niche, specialized, or low-resource languages or domains. Nevertheless, the assumption only comes into play as to our knowledge there is no specifically German trained BART model available. Unifying various subtasks on a data set might not always improve performance, but investigating transferability of this paradigm to other similar (German) tasks consisting of related subtasks seems worthwhile and promising.

8 Ethical Considerations

We acknowledge the potential concern about large-scale analysis of user content posted online. We argue that this issue applies only to a lesser extent to our approach, since the source of the content is mainly social networks, microblogs, news sites, and QA sites, which are explicitly written to be public. Furthermore, the only metadata available are the URLs such that the identities of the participants are not disclosed, as no personally identifiable data are collected. The data set utilized had previously already been collected, analyzed, and published.

Acknowledgements

This work is partially supported by the MOTIV research project funded by the Bavarian Research Institute for Digital Transformation (bidt), an institute of the Bavarian Academy of Sciences and Humanities. The authors are responsible for the content of this publication.

References

Matthias Aßenmacher, Alessandra Corvonato, and Christian Heumann. 2021. Re-evaluating germeval17 using german pre-trained language models. *CoRR*, abs/2102.12330.

Gregory W. Corder and Dale I. Foreman. 2014. *Nonparametric statistics: a step-by-step approach*, second edition edition. Wiley, Hoboken, New Jersey.

Olive Jean Dunn. 1964. Multiple Comparisons Using Rank Sums. *Technometrics*, 6(3):241–252.

Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Edouard Grave, Michael Auli, and Armand Joulin. 2020. Beyond English-Centric Multilingual Machine Translation. ArXiv:2010.11125 [cs].

Leonard Hövelmann and Christoph M. Friedrich. 2017. Fasttext and Gradient Boosted Trees at GermEval-2017 on Relevance Classification and Document-level Polarity. In *Proceedings of the GermEval 2017 – Shared Task on Aspect-based Sentiment in Social Media Customer Feedback*, pages 30–35, Berlin, Germany.

Ji-Ung Lee, Steffen Eger, Johannes Daxenberger, and Iryna Gurevych. 2017. UKP TU-DA at GermEval 2017: Deep learning for Aspect Based Sentiment Detection. In *Proceedings of the GermEval 2017 – Shared Task on Aspect-based Sentiment in Social Media Customer Feedback*, pages 22–29, Berlin, Germany.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Pruthwik Mishra, Vandan Mujadia, and Soujanya Lanka. 2017. GermEval 2017 : Sequence based Models for Customer Feedback Analysis. In *Proceedings of the GermEval 2017 – Shared Task on Aspect-based Sentiment in Social Media Customer Feedback*, pages 36–42, Berlin, Germany.

Behzad Naderalvojoud, Behrang Qasemizadeh, and Laura Kallmeyer. 2017. HU-HHU at GermEval-2017 Sub-task B: Lexicon-Based Deep Learning for Contextual Sentiment Analysis. In *Proceedings of the GermEval 2017 – Shared Task on Aspect-based Sentiment in Social Media Customer Feedback*, pages 18–21, Berlin, Germany.

Jan Pfister, Sebastian Wankerl, and Andreas Hotho. 2022. SenPoi at SemEval-2022 task 10: Point me to your opinion, SenPoi. In *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*, pages 1313–1323, Seattle, United States. Association for Computational Linguistics.

Raghav R, Adarsh Vemali, and Rajdeep Mukherjee. 2022. ETMS@IITKGP at SemEval-2022 task 10: Structured sentiment analysis using a generative approach. In *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*, pages 1373–1381, Seattle, United States. Association for Computational Linguistics.

Zeeshan Ali Sayyed, Daniel Dakota, and Sandra Kübler. 2017. IDS IUCL: Investigating Feature Selection and Oversampling for GermEval2017. In *Proceedings of*

Table 8: Configuration search space for our hyperparameter optimization conducted.

Parameter	Values
Model	facebook/bart-large-mnli, facebook/mbart-large-50, Shahm/bart-german, facebook/m2m100_418M
Batch Size	4, 8, 16
Epochs	1, 2, 3, 4, 5
Learning Rate	5e-6, 2.75e-5, 5e-5, 7.75e-05, 2.75e-04, 5e-4

the GermEval 2017 – Shared Task on Aspect-based Sentiment in Social Media Customer Feedback, pages 43–48, Berlin, Germany.

Thomas Scialom, Paul-Alexis Dray, Sylvain Lamprier, Benjamin Piwowarski, and Jacopo Staiano. 2020. MLSUM: The multilingual summarization corpus. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8051–8067, Online. Association for Computational Linguistics.

Yuqing Tang, Chau Tran, Xian Li, Peng-Jen Chen, Naman Goyal, Vishrav Chaudhary, Jiatao Gu, and Angela Fan. 2020. Multilingual translation with extensible multilingual pretraining and finetuning.

Michael Wojatzki, Eugen Ruppert, Sarah Holschneider, Torsten Zesch, and Chris Biemann. 2017. GermEval 2017: Shared Task on Aspect-based Sentiment in Social Media Customer Feedback. In *Proceedings of the GermEval 2017 – Shared Task on Aspect-based Sentiment in Social Media Customer Feedback*, pages 1–12, Berlin, Germany.

Hang Yan, Junqi Dai, Tuo Ji, Xipeng Qiu, and Zheng Zhang. 2021. A unified generative framework for aspect-based sentiment analysis. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2416–2429, Online. Association for Computational Linguistics.

A Hyperparameter Analysis

A.1 Parameter Range

In Table 8 we depict the range of explored hyperparameters, as described in Section 5.1. Since training runs for the largest learning rate (5e-4) resulted in training instabilities, the exploration of this learning rate was canceled early after no valid run could be conducted.

Table 9: Ranking of the selected hyperparameter combination across all subtasks.

	A	B	C1	C2	D1	D2
Rank	34	22	3	3	1	1

A.2 Ranking across Subtasks

As described in Section 5.1 we select the combination "facebook/mbart-large-50" as the best model with a learning rate of 5e-5, when training for 5 epochs and a batch size of 8. Table 9 lists the ranking of this selected hyperparameter configuration across all subtasks relative to all combinations of hyperparameters explored.

A.3 Results per Subtask

In the following we show the hyperparameter search results for subtasks A, B, C1, C2 and D2 on the validation set (Figures 4 to 8). As before, we consider $p < 0.05$ as statistically significant difference and mark it with an $*$. Black lines are again used to represent the average values. Overall, we find very similar results to subtask D1 (Section 5.1 and figure 3).

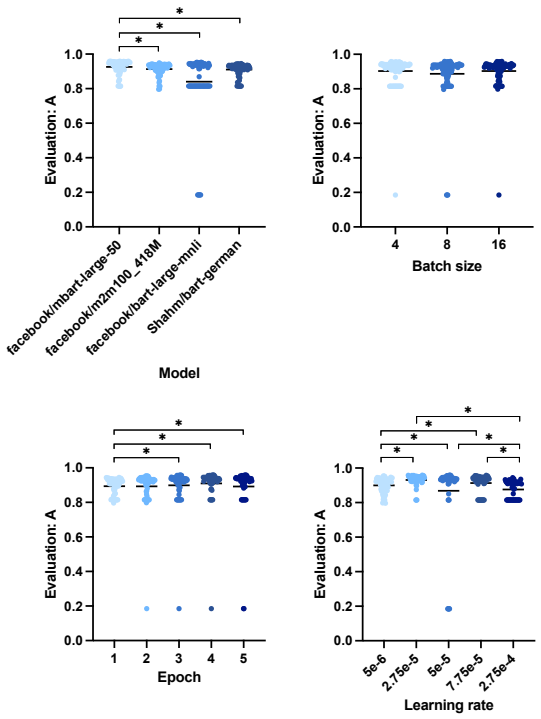

Figure 4: Results for subtask A, aggregated for each hyperparameter configuration. The hyperparameters are listed on the x-axis and the results for D1 on the y-axis.

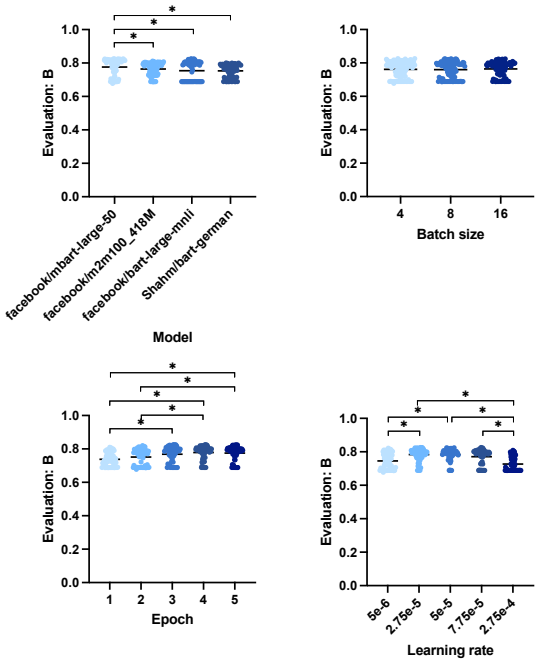

Figure 5: Results for subtask B, aggregated for each hyperparameter configuration. The hyperparameters are listed on the x-axis and the results for D1 on the y-axis.

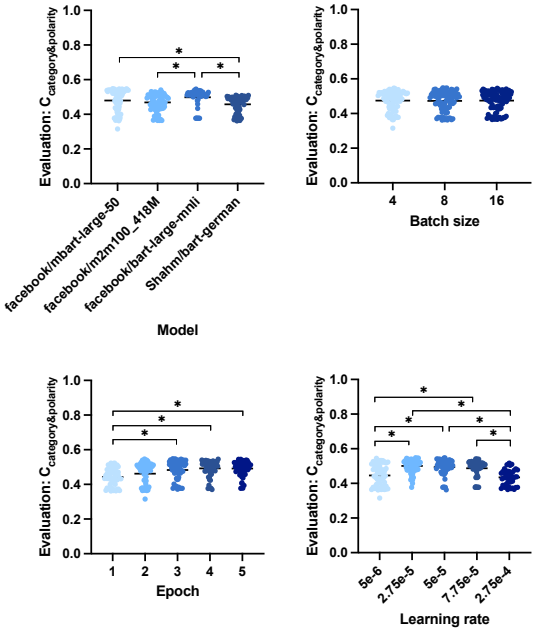

Figure 7: Results for subtask C2, aggregated for each hyperparameter configuration. The hyperparameters are listed on the x-axis and the results for D1 on the y-axis.

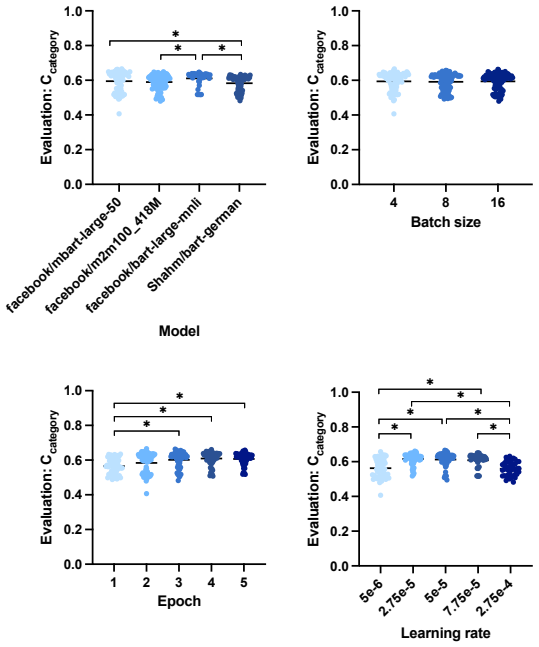

Figure 6: Results for subtask C1, aggregated for each hyperparameter configuration. The hyperparameters are listed on the x-axis and the results for D1 on the y-axis.

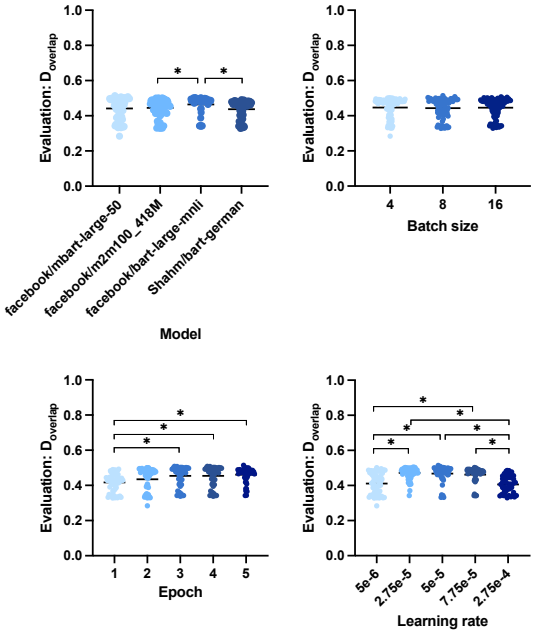

Figure 8: Results for subtask D2, aggregated for each hyperparameter configuration. The hyperparameters are listed on the x-axis and the results for D1 on the y-axis.

Table 10: Achieved results on the Test_{syn} set using different seeds. $\overline{x}$ denotes the average and σ the standard deviation. Both are computed per subtask.

Seed	A	B	C1	C2	D1	D2
1	0.940	0.807	0.594	0.485	0.371	0.388
2	0.954	0.832	0.635	0.525	0.419	0.443
3	0.951	0.829	0.633	0.527	0.418	0.444
4	0.942	0.816	0.618	0.507	0.407	0.430
5	0.948	0.821	0.633	0.516	0.413	0.436
42	0.953	0.815	0.632	0.510	0.415	0.440
$\overline{x}$	0.948	0.820	0.624	0.512	0.407	0.430
σ	0.006	0.009	0.016	0.015	0.018	0.021

B Seeds

When conducting the hyperparameter optimization we used the seed 42 for all runs. To illustrate how robust our results are to changes in random initialization, we additionally examined five more seeds in Table 10. It should be mentioned that, overall, the results are stable across different seeds. In our experiment, only seed value 1 can be viewed as an outlier, as it performs comparably poor.

C Analyzing Aßenmacher et al. (2021)

During the reevaluation of Aßenmacher et al. (2021), we found two performance differences when comparing their custom reimplementation of the metric with the original metric. All analyzes are based on the results generated using their repository: `https://github.com/ac74/reevaluating_germeval2017`. In the following we detail our analysis.

C.1 Subtask C

We reran subtask C and recorded the outputs obtained. During this run, according to the costum metric, the model achieved a performance close to their reported results. However, when we converted these results to the challenge XML format and evaluated it using the original GermEval binaries for easier comparison, we obtained the results, which we report in Table 6. We suspect different usages of micro vs. macro averaging to be the issue, but did not further investigate any possible differences between the metrics.

C.2 Subtask D

When inspecting the results obtained for subtask D, we observed that some spans in the input document are duplicated. This makes it hard to convert these predicted word-level span annotations to the original XML format, as the input string to be predicted gets changed during preprocessing. We show this using a truncated example: *"AZ Muenchen : Technischer Defekt: Störung am Isartor: S- Bahn-Stammstrecke dicht: Ein technischer Defekt[. . .]"*, which is annotated in the ground-truth with the opinion terms "Technischer Defekt" twice, as well as the terms "Störung" and "Bahn-Stammstrecke". During preprocessing this sample is transformed to the following input document: *"az muenchen : technischer defekt technischer defekt : storung am isartor : s - bahn - stammstrecke : storung am isartor : s - bahn - stammstrecke dicht : ein technischer defekt[. . .]"*, duplicating "technischer defekt" as well as "storung am isator: s-bahn-stammstrecke" in the process. Notably, these duplications seem to be connected to the ground-truth opinion spans. Thus, the annotations for the ground-truth label seem to leak into the model input, as spans that have to be annotated multiple times seem to be fed in multiple times. Due to this presumed bug in the data preprocessing, we are unable to reliably convert the model outputs to processable inputs for the original metric.